L11: Entry 1 of 3 File: PGPB Sep 18, 2003

DOCUMENT-IDENTIFIER: US 20030177187 A1
TITLE: Computing grid for massively multi-player online games and other multi-user immersive persistent-state and session-based applications

Summary of Invention Paragraph (5):
[0005] In recent decades, there has been rapid growth in the numbers of computers, and thus people, connected to the Internet, a vast network of computers connected by common communication protocols and data formats, and the World-Wide Web (WWW), a layer of structured information transmitted over the Internet. This increase of connectivity has allowed computer users to access various types of information, disseminate information, be participate in electronic commerce transactions, as well as engage in various forms of social interaction and entertainment previously limited by geographic and/or socio-political bounds.

Summary of Invention Paragraph (10):
[0010] Within the field of Concurrent Engineering, the state of the art tends to provides only loose integration between the applications or subsystems that provide multi-user interaction, the applications or subsystems that provide immersive simulation and applications or subsystems that collect data from sensors or otherwise interface with real-world processes and operations. While collaborative systems exist that allow engineers to exchange data, and to work on those data together, the majority of these systems are designed to merely transfer data files. Meta-information about the relationship of those files is stored (so that an interrelationship can be developed) in systems that are often termed "knowledge-based." These systems aid in the management and development of large projects, but they do not provide a uniform or holistic view of the component data. The interactions of users with those data are through multiple client programs, with no application providing a view of the whole. Interaction among and between users of the system tends to be "out of band," i.e., via email, instant messaging, Web-based discussion forums, etc. These communication systems can be bundled into an application suites, but the interactions take place outside the environment of the data models (the Simulations) themselves.

Summary of Invention Paragraph (11):
[0011] Visualization systems for collaborative work also exist. In general, these systems are data-file view utilities that allow users to view models produced by various client software programs with a single program. Additionally, they may allow users to annotate the files or modify them in some way, but they often do not allow the users to change those data to the same extent as the original authoring tools allow. These systems are beneficial in that users need not master the intricacies of multiple authoring tools to view different types of models, but again, they are not interactive.

Summary of Invention Paragraph (12):
[0012] Product and process life-cycle management systems (e.g., project management systems) are another important area of multi-user systems. These systems allow users to oversee the complete life-cycle, from conception to decommissioning of a product or system, including the design, manufacturing and operation of the product. Unfortunately these systems tend not to be closely integrated with the systems that are actually used to perform these discrete phases. They allow users to manage the system to an extent (by providing an overview of the program). Life-cycle management systems can also suffer from a common shortcoming in that real-time input that is germane to the operation of the program does not update the data model in real-time.

Summary of Invention Paragraph (17):
[0017] Another common characteristic (and short-coming) among the various multi-person, interactive applications is that the user (client) interface to the server-based

virtual environment is typically a personal computer, workstation or terminal where the user must distinguish between the real world and the virtual world. Consequently, users of multi-person interactive environments employ terms such as IRL ("in real life") to distinguish between their actual physical location (e.g., "I'm in my bedroom IRL."), and the virtual world (e.g., "I'm in the living room") which suggests that such a user is in the living room in the MMOG interactive application program, and not in the living room of their real house. In addition, the various multi-person interactive applications is that users cannot interact or otherwise respond to events that occur in the virtual (or real) environment when they are away from their personal computers, workstations or terminals. That is, users can not participate in the virtual, interactive, multi-person environment unless they are sitting at the computer. A further shortcoming is that due to their design and inability to cross technical platforms, current interactive applications are limited to a few client bases.

Detail Description Paragraph (83):
[0170] The present invention is primarily described in terms of a gaming example. This is for convenience only and is not intended to limit the application of the present invention. After reading the following description, it will be apparent to one skilled in the relevant art(s) how to implement the following invention in alternative embodiments (e.g., multi-user interactive applications focused on entertainment, simulations, project management, e-commerce, collaborative engineering, etc.). For example, in an alternate embodiment, a computer-aided design (CAD) application program executes within the Grid while maintaining referential integrity between a real life (physical) environment (e.g., a field engineer) and a computer-generated (synthetic) environment (e.g., a remotely-located designer using a CAD program). This allows the creation of synthetic models based on physically-derived (or observed) data, the maintenance and enhancement of synthetic models as change occurs in the physical world, and most importantly, the real-time interaction between physical and synthetic entities (e.g., persons).

Detail Description Paragraph (102):
[0189] For example, in one embodiment of the present invention, translator 108 may be a Web server which sends out Web pages in response to Hypertext Transfer Protocol (HTTP) requests from remote browsers (e.g., desktop computers 112f). The Web server would provide the "front end" to the users of the present invention. That is, the Web server would provide the graphical user interface (GUI) to users of Grid system 100 in the form of Web pages. Such users may access the Web server at the Multi-User Bridging organization's site via the transportation network 103 (e.g., the Internet and thus, the World Wide Web).

Detail Description Paragraph (119):
[0206] Database 104 stores the various types of information that Grid system 100 would need to store in order to provide the bridging of activities in real and virtual environments in the context of multi-user gaming, entertainment and e-commerce applications. Such information, includes user registration information (name, address, billing information, etc.), device 112 registrations, device 112 capabilities (e.g., polygon rendering capability, media formats, operating systems, available peripherals, color versus black-and-white display, etc.), user permissions (e.g., who is allowed to access portions of the bridged environment and what actions they may perform on those parts) and user ownership of synthetic entities and environment objects, entity location information, game environments, game rules, themes and roles, etc., as will be apparent to one skilled in the relevant art(s) after reading the teachings herein.

Detail Description Paragraph (122):
[0209] As will be appreciated by one skilled in the relevant art(s), whether application database 104 is an object, relational, and/or even flat-files depends on the character of the data being stored by the ASP which, in turn, is driven by the specific interactive, multi-user applications being offered by the ASP. Server 102 includes specific code logic to assemble components from any combination of these database models and to build the required answer to a query. In any event, translator 108, client devices 112, and/or administration workstation 106 are unaware of how, where, or in what format such data is stored.

Detail Description Paragraph (210):
[0297] The Gateway 401 provides an interface between the end-user device and other Game Servers 405. Note that in many cases this is not necessary; the Game Server 405 will work with generic UDP (User Datagram Protocol)/IP connections, and many client devices are capable of making and using these connections. In general, it is the lower-end platforms which will require a specialized version of the Gateway 401 to allow them to

interoperate. A WAP phone, for example, needs two levels of translation to interoperate with network-connected devices: a WAP Gateway to translate its native protocol to TCP/IP, and a WML server to format requests and displays in a form which can be displayed by the device. Players can use their service provider's Gateway, but the WML requests may be translated into a more generic network protocol by which the process servers operate.

Detail Description Paragraph (229):
[0316] FIG. 14 illustrates the login process in flow chart form, with the arrows designating process flow, showing the process of logging in, authentication and embodying one's Avatar. FIG. 14 should also be viewed in conjunction with other figures describing the Gateway 401 and the figures illustrating the Game Server 405 (see description below).

Detail Description Paragraph (288):
[0375] It is preferable to avoid designing Locales that are too small (room sized), too large (metropolitan sized) or too congested about the periphery (such as a park bounded by city streets). Care taken in intelligent design will go a long way to make the player's experience more enjoyable, with less lag and more rapid response times. Preferably, the Locale should be designed on the model of the "Locale region," on the order of magnitude of a few buildings or a city block with limitations on the ways in which traffic can logically enter or leave the region, as shown in FIG. 21. These recommendations should only be taken as a general guideline.

Detail Description Paragraph (370):
[0457] The payload is game data formatted in a particular way. The Network Protocol Stack is able to validate the format of individual packet payload without knowing or caring what the contents actually represent. The invariant properties of packets are the means that allow the syntactic validation (is the data "well-formed"?) of packet payload without requiring semantic validation (is the data meaningful?) below the level of the game itself.

Detail Description Paragraph (381):
[0468] If the Grid were not context agnostic, it might be reasonable to assume that the format of the data blocks could be left completely free and unrestricted. However, at a game system level, it is important to recognize the need for interoperability and extensibility. Thus, the Block Data is used to marshal object state throughout the Grid.

Detail Description Paragraph (382):
[0469] Referring again to FIG. 39, the format of a data block may include:

Detail Description Paragraph (387):
[0474] With these additional restrictions on the format of the payload block data, the Network Protocol Stack can perform its job quickly and efficiently. The NPS can receive, transmit, and validate packets. It can discriminate between essential and non-essential data; it can request retransmission of data that has become lost or corrupted in transit. It can guarantee that only properly versioned and formatted packets are forwarded to the game itself. It can do all this in a context agnostic manner, leaving the interpretation of the actual object state to the specific games that are up and running in the current environment.

Detail Description Paragraph (469):
[0556] FIG. 45 is an illustration of how the terms "region of interest", "region of presence," "personal space", etc. are used throughout this discussion and in particular as they relate to Dead Reckoning. FIG. 45 should be viewed in conjunction with FIGS. 46 and 47, and is also discussed below in the Area-of-Interest Management section.

Detail Description Paragraph (472):
[0559] FIG. 48 illustrates the dynamic interaction between two players located on different Locales and/or different Game Servers 405. In FIG. 48, Player 0 moves from right to left, as shown by the dotted line. The tag S0.L0.ER0.T0.0 in the figure refers to the following: S0 refers to Game Server 0, L0 refers to Locale Thread 0, ER0 refers to Embodiment of Record 0, and T0.0 refers Time0.0. The other tags in FIG. 48 have a similar format. Player 1 is a "white figure against a black background", and Player 2 is a "black figure against a white background", initially at Locale 0, Server 1. The two Embodiments of Record gradually approach each other such that their regions of interest intersect. The circle around Player 0, for example, is the region of interest around the Embodiment of Record 0 of Player 0. When the Embodiment of Record 0 moves to

a point where its region of interest touches the region of interest of Player 1 (i.e., of Embodiment of Record 1), a message is sent to Embodiment of Record 1, notifying it of that fact, and vice versa. Thus, this is how the Embodiment of Record 1 "sees" Embodiment of Record 0 walking towards it. In other words, Thing 0 is new to Thing 1, and a message needs to be propagated to reflect that fact.

Detail Description Paragraph (504):
[0591] c) Passing Python parameters as sub-blocks of type PYTHON :: GUID, PYTHON :: LONG, PYTHON :: FLOAT, PYTHON :: VECTOR, PYTHON :: ENUM or PYTHON :: STRING are optional and vary depending upon which function is invoked. The provided parameters will be packed and passed with a format string to the function itself before being executed on the server. It is the game designer's responsibility to decide which parameters are expected by each function, and in what order the parameters are to be provided.

Detail Description Paragraph (511):
[0598] def buy_a duck(format, parameters):

Detail Description Paragraph (512):
[0599] that requires two arguments, a format argument and a parameters argument. All rules enforcement script functions take these two arguments exactly. The format argument is a text string that, using special control characters, describes the order and type of the parameters that are packed into the second text string argument.

Detail Description Paragraph (514):
[0601] args=unpack(format, parameters)

Detail Description Paragraph (597):
[0684] In an alternative embodiment, on more modest client device 112 platforms, this layer may be complex and could involve "lossy" translations, where certain data-elements are parsed out and not transmitted to the end-client. As will be appreciated by those skilled in the relevant art(s), "lossy" is a term describing a data compression algorithm that actually reduces the amount of information in the data, rather than just the number of bits used to represent that information. The lost information is usually removed because it is subjectively less important to the quality of the data (usually an image or sound) or because it can be recovered reasonably by interpolation from the remaining data. The JPEG and MPEG formats are lossy algorithms.

Detail Description Paragraph (607):
[0694] In one example, the system 100 of the present invention is implemented in a multi-platform (platform independent) programming language such as JAVA, programming language/structured query language (PL/SQL), hyper-text mark-up language (HTML), practical extraction report language (PERL), common translator interface/structured query language (CGI/SQL) or the like. Java-enabled and JavaScript-enabled browsers are used, such as, Netscape, HotJava, and Microsoft Explorer browsers. Active content Web pages can be used. Such active content Web pages can include Java applets or ActiveX controls, or any other active content technology developed now or in the future. The present invention, however, is not intended to be limited to Java, JavaScript, or their enabled browsers, developed now or in the future, as would be apparent to a person skilled in the relevant art(s) given this description.

Detail Description Table CWU (3):
3 #################### begin python example code #!/usr/Locale/bin/python # butterfly.py - example python script # import sys import types from struct import * from server import * #all parameters to python functions are passed #as a format string, followed by the packet parameters. . . #use the utility "unpack" to extract these parameters into #an argument list for processing by the python code. . . #arguments passed to python routine "buy_a_duck" # #arg0--caller GUID (passed in by system) #arg1--GUID of the particular duck to buy #arg2--Thing_type of duck (animal type) #arg3--GUID of prospective purchaser of the duck #arg4--Thing_type of purchaser (Avatar type) #arg5--PropertyID of the purchaser's inventory list def buy_a_duck(format,parameters): args = unpack(format,parameters) sys.stderr.write("python- .buy_a_duck%s\n" % str(args)) # check there are enough args and they are of correct types if len(args) > 5.backslash. and
isinstance(args[1],types.IntType).backslash. and
isinstance(args[2],types.IntType).backslash. and
isinstance(args[3],types.IntType).backslash. and
isinstance(args[4],types.IntType).backslash. and
isinstance(args[5],types.IntType).backslash. # properties as arguments to. . .ByGUID( )

```
# methods are passed in CTHINGATTRIBUTEVALUEBUFFER value = CThingAttributeValueBuffer(
) value.m_Attribute.Type = PROPERTY_STRING value.m_typeObject = 0 value.m_bDirty = 0
value.bufferString(17, "wanna buy a duck?") # ask the purchaser if they want to buy the
duck # this may generate a secure dialog with the user askApprovalByGUID(args[3],value)
# askApproval returns GUID of authorized purchaser if(value.m_Attribute.Type !=
PROPERTY_LONG).backslash. or not(value.m_bDirty): sys.stderr.write.backslash. ("need
authorisation to buy duck %d.backslash.n".backslash. % args[1]) return # if we make it
this far we have received approval # from the prospective purchaser of the duck (arg3)
sys.stderr.write("got approval %d " % value.m_bDirty) sys.stderr.write("from guid
%d.backslash.n".backslash. % value.m_Attribute.Value.lLong) # the grabByGUID( ) method
attempts to stuff the duck # into the purchaser's inventory list: it returns the #
former location of the duck if the operation succeeds. value.m_Attribute.Type =
PROPERTY_VECTOR value.m_idState = POSITION value.m_typeObject = 0 value.m_bDirty = 0
value.m_Attribute.Value.vVector.x = 0 value.m_Attribute.Value.vVector.y = 0
value.m_Attribute.Value.v- Vector.z = 0 grabyGUID.backslash. args[1], args[2],
args[3],args[4], args[5], value) # check the resulting value for the former location
and # print out the result of this secure transaction. . .. if value.m_Attribute.Type
!= PROPERTY_VECTOR: sys.stderr.write("failed to buy duck %d.backslash.n" .backslash. %
args[1]) else: sys.stderr.write("bought duck %ld " % args[1]) sys.stderr.write("located
at %f" % .backslash. value.m_Attribute.Value.vVector.x) sys.stderr.write(", %f" %
.backslash. value.m_Attribute.Value.vVector.y) sys.stderr.write(", %f" % .backslash.
value.m Attribute.Value.vVector.z) sys.stderr.write(".backslash.n") return
##################### end python example code
```

WEST

Generate Collection Print

L22: Entry 4 of 7 File: USPT May 11, 1999

DOCUMENT-IDENTIFIER: US 5903886 A
TITLE: Hierarchical adaptive state machine for emulating and augmenting software

Abstract Text (1):
The present invention defines a method for emulating an iterated process represented by a series of related tasks and a control mechanism that monitors and enables the iterative execution of those tasks until data associated with the process converges to predetermined goals or objectives. The invention defines a method in which fuzzy neural networks and discreet algorithms are applied to perform the process tasks and in which configurable, reloadable finite state machines are applied to control the execution of those tasks. In particular, the present invention provides a method for emulating the process of designing integrated circuit (IC) applications and printed circuit board (PCB) applications for the purpose of simulating, emulating, analyzing, optimizing and predicting the behavioral and physical characteristics of the application at the earliest possible stage of the process. The invention applies fuzzy neural networks and configurable, reloadable finite state machines to emulate the IC or PCB design process, enabling the invention to emulate the the computer aided design (CAD) tools used to perform the design process tasks as well as the individuals using those tools. By emulating the combination of man and machine performances, the invention can more accurately predict the results of a given task than tools that consider only the machine element. The invention also provides a means to adapt the performance and behavior of any element of the invention using historical data compiled from previous design or manufacturing experiences, allowing the invention to incorporate the knowledge gained from previous designs into current designs.

Detailed Description Text (57):
The Schematic Parser Function, is simply an interface as defined by a commercially available tool used for Schematic Capture and a parser. The interface can take the form of some customization language, like AMPLE.TM. for Mentor Graphics' Design Architect schematic capture or SKIL.TM. for Cadence's Compose schematic capture, or a standard data format, like EDIF (Electronic Design Interchange Format) The parser section of Function B will take this raw input and translate it into a data format representing the internal standard by which the invention keeps its internal data. This is a simple matter of building one data structure from another.

Detailed Description Text (139):
The process of designing Printed Circuit Board (PCB) applications is very similar to the IC design process, requiring the same process steps like capture, placement, routing, and analysis, among others. By using sets of data compiled for the PCB design process to train the Pseudo Placement, Pseudo Route, and relevant Calculator virtual tools, an alternate set of applications or a single application is enabled which emulates the PCB design process for the purpose of simulating, emulating, analyzing, optimizing, or predicting interconnect noise and/or crosstalk data during the capture phase of the design process.

Detailed Description Text (140):
Additionally, the process of designing gate array, standard cell, and other application specific integrated circuits (ASICs) is the same as the IC process described in the preceding paragraphs. By using sets of data compiled for the ASIC design process to train the Pseudo Placement, Pseudo Route, and relevant Calculator virtual tools, an alternate set of applications or a single application is enabled which emulates the ASIC design process for the purpose of simulating, emulating, analyzing, optimizing, or predicting interconnect noise and/or crosstalk data during the capture phase of the design process.

WEST

Freeform Search

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L63 and (access$ or retriev$ or transmit$)

Display: 50 Documents in Display Format: - Starting with Number 1

Generate: ○ Hit List [✓ Hit Count] ○ Side by Side ○ Image

Search | Clear | Help | Logout | Interrupt

Main Menu | Show S Numbers | Edit S Numbers | Preferences | Cases

Search History

DATE: Friday, October 31, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L67	L64 and library	6	L67
L66	L64 and (read and verifier)	0	L66
L65	L64 and (read$ and verify$)	6	L65
L64	L63 and (access$ or retriev$ or transmit$)	8	L64
L63	L62 and properties	8	L63
L62	L61 and (format or rules)	11	L62
L61	L60 and (data near model)	11	L61
L60	(707/$.ccls.) and (CASE near design)	41	L60
L59	L58 and CASE	14013	L59
L58	((707/$)!.CCLS.)	17235	L58
L57	L56 and (annotation near file)	0	L57
L56	L54 and parser	31	L56

L55	L54 and (design near file)	0	L55
L54	L52 and script$	66	L54
L53	L52 script$	26873	L53
L52	L51 and (user near interface)	69	L52
L51	L50 and behavior	72	L51
L50	L49 and (software near application)	100	L50
L49	L47 and (search$ or quer$)	125	L49
L48	L47 and (format near writer)	0	L48
L47	L40 and import$	127	L47
L46	Ll40 and import$	2	L46
L45	L40 and (import near application)	0	L45
L44	L40 and (import near application n)	125	L44
L43	L40 and (import near application near grogram)	0	L43
L42	L40 and (import near application near programming)	0	L42
L41	L40 and (import near application near grogramming)	0	L41
L40	L39 and library	139	L40
L39	L38 and (application near program)	156	L39
L38	L37 and format	217	L38
L37	L35 and properties	241	L37
L36	L35 and (intelligent near design)	1	L36
L35	L34 and (data near model)	323	L35
L34	L33 and (single near application)	6310	L34
L33	(access$ or retriev$ or transmit$) and CASE	1075107	L33
L32	(access$ or retriev$ or transmit$) and l24	1075116	L32
L31	L30 and (application near program)	3	L31
L30	L29 and format	8	L30
L29	L28 and properties	10	L29
L28	L27 and (intelligent near design)	10	L28
L27	L26 and (CASE)	4350	L27
L26	L25 and (data near model)	4351	L26
L25	L24 and design	647914	L25
L24	CASE or (computer near aided near software near engineering)	3192082	L24
L23	(data near model) and properties and case	3742	L23
L22	L21 and (single near application)	7	L22
L21	electronic near design and CAD	603	L21
L20	L19 and (single near application)	5	L20
L19	(access$ or retrieve$ near tramit$) and (electronic near (document or product) near design)	94	L19
L18	L15 and (single near application)	5	L18
L17	L15 and (single near application)	5	L17

L16	L15 and (singlr near application)	0	L16
L15	(access$ or retriev$ or transmit$) and (electronic near product near design$)	167	L15
L14	(access $ or retriev$ or transmit$) and (electronic near product near design$)	234	L14
L13	(single near application) and (access$ or transmit$) and (electronic near product)	24	L13
L12	(single near application) same (access$ or transmit$) same (electronic near product)	0	L12
L11	L10 and (data near model)	3	L11
L10	L9 and (application near program)	28	L10
L9	L8 and format	80	L9
L8	L1 and (view$ or brows$)	183	L8
L7	L1 and ((reader and verifier) near format)	2	L7
L6	intelligent near design near data	2	L6
L5	L4 and (format near reader)	1	L5
L4	L3 and (format near verifier)	1	L4
L3	L2 and format$	46	L3
L2	L1 and library	56	L2
L1	intelligent near design	249	L1

END OF SEARCH HISTORY